AFS

PTO/SB/21 (02-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/019,827
Filing Date	5/21/2002
First Named Inventor	Luo, et al
Art Unit	2625
Examiner Name	Manev Seth
Total Number of Pages in This Submission: 19	Attorney Docket Number: fraunh01.028

ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
- ☑ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance communication to Technology Center (TC)
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☑ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☑ Other Enclosure(s) (please Identify below):

Return Postcard

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Gordon E. Nelson #30,093
Signature	/Gordon E. Nelson/
Date	10/11/2006

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	Gordon E. Nelson #30,093		
Signature	/Gordon E. Nelson/	Date	10/11/2006

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
(fraunh01.028)

Applicant: Luo, et al. **Confirmation No.: 8660**

Application No: 10/019,827 **Group Art Unit:** 2625

Filed: 5/21/02 **Examiner:** Manev Seth

Title: *Authenticating executable code and executions thereof*

Commissioner for Patents
Alexandria, VA 22313-1450

Appeal Brief under 37 C.F.R. 41.37

(1) Real party in interest

The real party in interest is Fraunhofer, CRCG, which is a subsidiary of Fraunhofer Gesellschaft, Munich, Germany.

(2) Related appeals and interferences

None.

(3) Status of claims

Claim 2 has been canceled; claims 1, 3-28 have been rejected.

(4) Status of amendments

An amendment after final was filed on 10/3/06 in which claim 10 was amended to make it dependent from claim 8 instead of claim 9. That amendment has been entered.

(5) Summary of claimed subject matter

Overview of the claimed subject matter

The independent claims in this application are claims 1, 18, and 21; the subject matter of all of these claims is techniques for *authenticating* executable code by watermarking the executable code with a hidden watermark and then using the watermark to determine whether the code is authentic. The meaning of the term "authenticate" is critical to the proper understanding of Applicants' claims. The term is defined as follows at page 10, lines 1-2 of Applicants' Specification.

> Digital representations are *authenticated* to make sure that they have not been altered in transit.

As pointed out beginning at page 5, line 11 through page 6, line 16 of Applicants' Specification, the rapidly increasing use of mobile code has increased the need for effective techniques for authenticating executable code. Mobile code is code which is downloaded to a device attached to a network in the course of an interaction between a user of the device and the network (or another device attached to the network) and is then executed as part of the interaction. Mobile code is ubiquitous in the Internet. Many Web pages include mobile code written in the Java™ or ActiveX programming languages. When the Web page is received in a browser, the mobile code is executed by the computer upon which the browser is written. Mobile code is also used to implement features in devices such as cellular telephones. When a user does something with the cellular telephone which requires the feature, mobile code for the feature is downloaded to the cellular telephone and then used in the interactions that involve the feature.

While mobile code is useful, it can be dangerous both to the system that receives the code and to the system that provides the code for downloading. The danger to the receiving system is that the code is not what it appears to be: it may have been modified to include a virus or an Internet worm that can damage the receiving system or it may have been modified to return different or additional data or to return the data to a different location. The danger to the sending system is that the code that is being executed is not the code that was sent. When the sending system is legitimate, it does not want the receiving system to receive code that appears to come from the sending system but has been modified to include a virus or to otherwise change the code's behavior. When the sending system receives data from an execution of the mobile code on the receiving system, the sending system needs to be sure that the data is coming from an execution of the code that the sending system provided to the receiving system. Authentication permits both

the receiving system to be sure that the code it received is what the sending system actually sent and the sending system to be sure that the data it is receiving is from an execution of the mobile code that it sent to the receiving system.

Digital *watermarks* are ordinarily used to hide information in digital representations. They are typically employed to hide information such as the copyright owner's name in digital representations of images, audio, or video. When used in digital representations of images, audio, or video, the watermark can be hidden in noise in the digital representation. Watermarking executable code is a more difficult problem, because executable code does not contain noise. A number of techniques for watermarking executable code have, however, been developed. The techniques are summarized at page 6, line 18 through page 7, line 15 of Applicants' *Description of related art.* The techniques fall into two classes: *static techniques*, in which the watermark is embedded in the executable code itself, and *dynamic techniques*, in which the watermark is embedded in properties of executions of the code. Many static and dynamic techniques for watermarking software are explained in detail in Collberg, et al., *Software watermarking techniques*, the chief reference applied by Examiner in his rejection of the claims.

The authentication technique is described in overview at pp. 20, line 20 through 22, line 7 and shown in FIG. 6. There are two parts to the technique: watermarking the executable code to make it authenticatable, shown at 601 and described at page 21, lines 4-20 and then using the watermark to authenticate the executable code, shown at 615 and described at page 21, line 21-22, line 7. In watermarking, executable code 603 is processed by a watermarker 605 which uses a key 609 to watermark the code with a watermark value 607 in any of the ways in which executable code can be watermarked. The watermarking technique being used in FIG. 6 is static watermarking. The watermarked executable code 613 is then made available to its destination along with key 609 and value 607, which may be encrypted. One way of making key 609 and value 607 available is to combine them in a code package 611 with watermarked executable code 613 and sending the code package to the destination.

At the destination, a watermark reader 617 uses key 609 to obtain a watermark value 623 from watermarked executable code 613. Comparator 621 then compares watermark value 607 with

obtained watermark value 623. If the two are sufficiently similar, watermarked executable code 613 has not been altered in transit and is authentic. What "sufficiently similar" means depends on the kind of watermarking used; with static watermarking, any change in the watermark value or the lack of a watermark value indicates that the code has been altered and is therefore not authentic.

Summary of the subject matter of the independent claims

As befits the nature of the invention, there are separate independent claims to watermarking code according to the invention and authenticating the watermarked code according to the invention. Claim 1 is addressed to watermarking code according to the invention and claims 18 and 21 are addressed to authenticating such watermarking code. Claim 1 is generic to both static and dynamic watermarking techniques; claim 18 is addressed to authentication using a static watermark and claim 21 is addressed to authentication using a dynamic watermark.

Summary of the subject matter claimed in claim 1

Claim 1 reads as follows. Reference numbers and citations to the Specification as filed have been inserted for the convenience of the Board but are not intended to limit the claim:

> **1.** A method of adding a watermark (607) to a sequence of executable instructions (603) to render the sequence authenticatable,
> the method comprising the steps of:
> receiving the sequence of executable instructions and a key (609)(p. 21, lines 6-8); and
> using the key to modify the sequence of executable instructions so that the watermark is obtainable from the modified sequence, the sequence being modified such that the usefulness of the modified sequence for the sequence's intended purpose is not affected by the modifications made thereto and the watermark representing a watermark value (p. 21, lines 8-11), alteration or absence of the watermark value being used when the sequence is authenticated to determine whether the sequence is authentic (p. 21, lines 28-30).

The reference numbers and locations cited in the claim are from the disclosure of static watermarking; as mentioned above, claim 1 is generic to both static and dynamic watermarking. Watermarking and authentication using dynamic techniques are shown in FIG. 9 and are described in overview at page 23, line 10 through page 24, line 26.

Claim 18

Claim 18 is addressed to authentication with static watermarking.

18. A method of authenticating a watermarked sequence of executable instructions (613) watermark having been produced by modifying the sequence according to a key 609) that certain of the instructions in the sequence represent a watermark value (607)
the method comprising the steps of:
receiving the watermarked sequence or a copy thereof (p. 21, lines 23-24)
using the key to locate the certain instructions in the received sequence and read the watermark value (p. 23, lines 24-26)
using alteration or absence of the watermark value to determine whether the received sequence is authentic (p. 23, lines 28-31).

Claim 21

Claim 21 is addressed to authentication with dynamic watermarking:

21. A method of authenticating a sequence of executable instructions that has been watermarked (907) by modifying the sequence according to a key (905) such that when the sequence is executed, first execution state (917) is produced,
the method comprising the steps of:
receiving a description of second execution state (909)(p. 24, lines 18-20; p. 25, lines 9-10); and
if the received description does not describe the first execution state, determining that the sequence of executable instructions whose execution produced the second execution state is not authentic (p. 24, lines 20-23; p. 25, lines 10-16).

Claim 21 is generic both to the case where an actual execution state is compared with expected state patterns to determine authenticity and the case where the actual execution state is compared with a description of the expected execution state.

(6) Grounds of rejection to be reviewed on appeal

The grounds of rejection to be reviewed are:

- The rejection of claims 1-18 under 35 U.S.C. 103(a) as being unpatentable over Collberg, et al., ACM Publication 1999, "Software Watermarking: Models and Dynamic Embeddings" (henceforth Collberg, *Software Watermarking)* in

combination with U.S. Patent 5,905,800, Moskowitz, et al., *Method and system for digital watermarking*, issued May 18, 1999, (henceforth Moskowitz '800).

- The rejection of claims 19-28 under 35 U.S.C. 103(a) as being unpatentable over the combination of Collberg, *Software Watermarking*, Moskowitz '800, and Collberg, Thomborson, and Low, *A Taxonomy of obfuscating transformations*, Technical report #148, Department of Computer Science, July, 1997 (henceforth Low).

There are at present no other grounds of rejection in the application.

(7) Argument

In general

As set forth at MPEP 2143, a rejection under 35 U.S.C. 103 requires that the examiner make a *prima facie* case of obviousness. One of the elements of the *prima facie* case is that the combination of references that forms the basis of the rejection show *all* of the limitations of the claim under rejection. The following discussions of the rejections of the claims will show that *none* of the references disclose the use of watermarks in executable code to authenticate the executable code. All of Applicants' claims include this limitation; consequently, if none of the references discloses it, Examiner has not made his *prima facie* case for any of the claims.

The disclosure of Collberg, *Software Watermarking*

As the title indicates, Collberg *Software Watermarking* does disclose watermarking software and the use of keys to do so. For Collberg, the purpose of the watermarking is to embed a copyright notice or customer identification number in the software (*Introduction*, first paragraph). There is simply no notion in Collberg, *Software Watermarking* that a watermark might be used to *authenticate* the software. The character string "authentic" does not appear in the reference. The closest that Collberg, *Software watermarking* comes to the topic of authentication is the discussions of tamperproofing watermarks in sections 2.3 and 5.5 and 5.5.1. Tamperproofing a watermark is taking measures to make sure that the watermark cannot be removed from the code and this, of course, is not the same as tamperproofing the code that carries the watermark. Section 5.5.1 does include one example which shows Java's class reflection

mechanism may be used to detect tampering with a watermark based on a graph's node type and the program may be made to terminate on detection.

What Collberg, *Software watermarking* is chiefly concerned with is the difficulty of watermarking something which is as malleable as executable code. For example, static watermarks in executable code can be removed simply by obfuscating the executable code until the watermark is no longer detectable (see Section 2.3). For that reason, Collberg, *Software watermarking* prefers dynamic watermarking, but even there, obfuscation can render most dynamic watermarks undetectable (see Section 5). In Section 5, Collberg, *Software watermarking* discloses a new watermarking technique called "dynamic graph watermarking" which has more resistance to obfuscation attacks. The conclusion which must be drawn from Collberg, *Software watermarking* is that only the most complex dynamic watermarking techniques are of any use in protecting executable code.

Watermarking and authentication

While Collberg's pessimism about watermarking executable code may be justified when the watermark is used to show ownership, Applicants have demonstrated that even simple static watermarking is an effective way to *authenticate* executable code. The reason that this is so is that in authentication, loss or corruption of the watermark is *proof that the code has changed since it was watermarked.* Thus, the very property of watermarked code that renders the watermark almost useless for showing ownership of the code makes the watermark extremely useful for detecting faulty transmission of the code or tampering with the code and therefore for authenticating the code.

Examiner's response to Collberg's failure to disclose the use of watermarks in executable code for authentication

First of all, Examiner admits, "Collberg does not provide very clear teachings and for the sake of better clarity that it's the watermark value that determines the authenticity of the program, examiner cites Moskowitz" (final Office action of 2/16/06, p. 9, lines 9-11).

Secondly, Examiner argues at page 8, lines 11-16 of the final Office action of 2/16/06 as follows:

> As it is well known that adding watermarks will provide copyright security to the programs and how susceptible a program is to be getting copied depends on how good is the water mark and therefore it apparently is the watermark value as disclosed by Collberg ("W has a mathematical property that allows us to argue that its presence in P is the result of deliberate actions") that provides the authenticity of the program or sequence.

This argument completely confuses two functions of watermarks: protection against copying and authentication checking. First, a watermark used to protect a file against copying does not render the file that contains the watermark uncopyable; all it does is insure that the copied file will contain ownership information such as the owner's name or a copyright notice. Second, what authentication is concerned with is not whether a program is an illegal copy, but rather whether the program has changed or been tampered with in transit.

Finally, Examiner cites locations in pages 314-317 of Collberg, *Software Watermarking* as showing the use of watermarks for authentication. The discussion at 315 is an explication of FIG. 2 of the reference (page 314, top), which shows techniques for embedding dynamic watermarks. Nothing in this discussion discloses anything about using watermarks to authenticate executable code. The same is the case with the discussion at page 316, which describes watermark extraction, the discussion at page 317, which describes how watermarks are susceptible to attacks that increase the static size of the code, or the discussion at page 314, last paragraph, which refers to another Moskowitz patent, U.S Patent 5,745,569, Moskowitz, *Method for stega-cipher protection of computer code*, issued 1996. Examiner does not cite this patent against Applicants, and what it discloses is not the use of a watermark to authenticate software, but rather a piracy prevention scheme in which an image file includes a watermark whose content is code that is necessary to display the image file. If the watermark is absent or damaged, the code cannot be extracted from the watermark and executed and the image file cannot be displayed.

It should be pointed out here that Applicants' techniques for determining the authenticity of software may employ watermarks made using Collberg's techniques. Because Collberg's watermarks can be used for authentication, it is striking that Collberg discusses software watermarking techniques at great length without even mentioning authentication. The only inference that can be drawn from this is that Collberg has absolutely no notion that watermarks in executable code might be useful for authenticating the code. Indeed, Collberg's general pessimism about the usefulness of watermarks for hiding identification information in software effectively teaches away from *any* use of watermarks to protect software or its users.

The disclosure of Moskowitz '800

What Moskowitz '800 discloses is well-described in its *Abstract*:

> A method for applying a digital watermark to a content signal is disclosed. In accordance with such a method, a watermarking key is identified. The watermarking key includes a binary sequence and information describing application of that binary sequence to the content signal. The digital watermark is then encoded within the content signal at one or more locations determined by the watermarking key.

Examples of a "content signal" are digital audio or video (col. 2, lines 39-40). As described above, the techniques used to watermark such "content signals" differ completely from those used to watermark software. The digital watermark carries information such as a copy's title, copyright holder, or the licensed owner of a particular copy (col. 1, lines 14-18). As would be expected from the *Abstract*, there is no disclosure whatever in Moskowitz of using watermarks to authenticate executable code or even of using watermarks to authenticate files containing the watermarked content signals.

Authentication is discussed in Moskowitz. However, as disclosed by a search on the stem "authentic" in Moskowitz '800, what is authenticated in Moskowitz is the information in the watermark itself (col. 2, lines 21-24, col. 7, lines 10-13, and col. 9, lines 44-46). Further, an individual purchase may be authenticated by decoding the copyright notice in the watermark. (col. 3, lines 1-5). None of this, of course, has

anything to do with authentication as that term is used in Applicants' specification or with watermarking executable code and then using the watermark to authenticate the executable code.

Examiner's application of Moskowitz

Given the disclosure of Moskowitz, it is not surprising that Examiner has difficulties applying the reference to Applicants' claims. In applying Moskowitz at page 6 of the final Office action of 2/16/06, Examiner states,

> Clearly Moskowitz's teaching provide that the original key which describes the same sequence will authenticate the user to access the sequence, or apparently in other words, if the sequence is tampered, the key won't work as the sequence won't be the same.

The first problem with this is that the function of Moskowitz' key is not the authentication of the user, but simply to locate the watermark in the content file. Second, what is authenticated in Applicants' invention is the software to the *user*, not the user to the *software*. The issue addressed by authentication is not whether a given user has the *right* to use the software, but rather whether the software that the user has received has been *altered* in transit.

Examiner further points at page 6 of the Office action to col. 1, lines 40-50 of Moskowitz '800, which sets forth that an advantage of using a key to hide a watermark is that a party who does not have the key cannot remove the watermark without damaging the watermarked content. Of course, when a file's content has been damaged, there is no need to use the watermark to determine whether the damaged file is authentic. This, however, has nothing to do with the authentication of code using watermarks. In the context of executable code, use of the watermark for authentication makes it possible to detect code which is perfectly valid executable code but which has been maliciously altered. See in this regard page 5, line 25-page 26, line 3 of Applicants' Specification.

Examiner finally rebuts Applicants' contention that Moskowitz adds nothing to Collberg concerning the use of watermarks to authenticate software by referring Applicants to col.

4, lines 40-55 and col. 9, lines 1-20 of the reference. Col. 4, lines 50-55 describe how human assistance "would allow for a better match of a given informational signal (be it an IRC code , an audio or voice file, serial number or other 'file' format) to the underlying content ...". Col. 1, lines 38-39 of the reference make it clear that what is meant by "informational signal" is the signal that carries the watermark's content; consequently, the location does not disclose watermarking of software. Col. 9, lines 1-20 disclose how watermarks may contain "information to be used in effecting software or content metering services" and sets forth the possible content of such information. Since the entire discussion of Moskowitz '800 concerns watermarking of content files which are not executable code, any watermark used to meter software is most probably located in content files belonging to the software rather than in the software's executable code. Be that as it may, the location certainly does not disclose using watermarks in executable code to *authenticate* the executable code.

The disclosure of Low

All that Low has in common with the watermarking references is that modifying code to add a static watermark or to produce a dynamic watermark and obfuscating code both result in modification of the code. The modifications are, however, for different purposes and done in different manners. Obfuscation as described in Low is done to make reverse engineering of the code harder (Low page 3, second column, paragraphs 3 and 4), while Applicants' modifications are done to watermark the code and thereby make it authenticatable. There is no indication whatever in Low that obfuscation as described there has anything whatever to do with watermarks or with authentication.

Examiner's application of Low

Examiner cites page 25, section 9.5 of Low as disclosing claim 18's step of comparing watermark values. What the cited location discloses is how a software engineer who is reverse engineering an obfuscated program can instrument the obfuscated program and then examine the run-time behavior of the obfuscated program to identify portions of the obfuscated program that were perhaps added by the obfuscation process and then test whether the portions were added by assigning a behavior to those portions, making a new

version of the program in which all of those portions are replaced by the behavior, and then seeing whether the original obfuscated program and the new version produce the same outputs. If they do, at least some of the obfuscating portions have been identified. Obviously, none of this has anything to do with watermarking executable code and using the watermarks to authenticate the executable code.

The rejections of the individual claims

The independent claims

Rejection of a claim under 35 U.S.C. 103 requires that the references used in the rejection show all of the limitations of the claim under rejection. Each of the independent claims 1, 18 and 21 include an authentication limitation; as set forth above, none of the cited references discloses using watermarks in executable code to authenticate the watermarked code; consequently, the combination of Collberg, *Software Watermarking* and Moskowitz '800 cannot serve as a basis for the rejection of claims 1 and 18 and the combination of Collberg, *Software Watermarking*, Moskowitz '800, and Low cannot serve as a basis for the rejection of claim 21. Since the independent claims are patentable over the references, so are all of the claims dependent from the independent claims.

The dependent claims

Certain of the dependent claims are patentable in their own rights over the references. The only reference which discloses anything about software watermarking is Collberg, *Software Watermarking*, and it does not disclose the use of the properties set forth in claims 11-14 for watermarking. Examiner rejects claims 11-14 on the basis of FIGs. 2, 3, and 5 and the discussion on pages 315-317 and at sections 5.1. and 5.3, but none of these locations disclose the use as a watermark value of "an order of elements in the output" (claim 11), "an additional element in the output" (claim 12), "a class of an element in the output" (claim 13), or "a constraint that is satisfied by elements of the output" (claim 14). The same is true with regard to dependent claims 25-28.

Conclusion

In the foregoing, Applicant has complied with the requirements of 37 C.F.R. 41.37 with regard to his brief and has demonstrated in the brief that examiner has failed to establish a *prima facie* case of obviousness with regard to *any* of his rejections under 35 U.S.C. 103. That being the case, the rejections cannot stand and Applicant respectfully requests that the Board of Appeals reverse the examiner with regard to all of his rejections and remand the application to the examiner for further processing as indicated by the reversals. A check for the appeal fee of $250.00 is attached. No other fees are believed to be required. Should any be, please charge them to deposit account #501315; any overpayment should be credited to that account.

Respectfully submitted,

/Gordon E. Nelson/
Attorney of record,
Gordon E. Nelson
57 Central St., P.O. Box 782
Rowley, MA, 01969,
Registration number 30,093
Voice: (978) 948-7632
Fax: (866) 723-0359
10/11/06
Date

Certificate of Mailing

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to:

Assistant Commissioner for Patents
Washington, D.C. 20231

on 10/11/06
(Date)
Gordon E. Nelson, #30,093
/Gordon E. Nelson/

(8) Appendix of claims

1. A method of adding a watermark to a sequence of executable instructions to render the sequence authenticatable,
the method comprising the steps of:
receiving the sequence of executable instructions and a key; and
using the key to modify the sequence of executable instructions so that the watermark is obtainable from the modified sequence, the sequence being modified such that the usefulness of the modified sequence for the sequence's intended purpose is not affected by the modifications made thereto and the watermark representing a watermark value, alteration or absence of the watermark value being used when the sequence is authenticated to determine whether the sequence is authentic.

2. (canceled)

3. The method set forth in claim 1 wherein the step of modifying the sequence includes the steps of:
using the key to determine locations in the sequence including modification locations at which the sequence is to be modified; and
modifying the sequence at the modification locations such that the locations specified by the key represent the watermark value,
whereby the watermark value is obtainable from the modification locations.

4. The method set forth in claim 3 wherein the step of modifying the sequence includes the step of:
inserting one or more executable instructions at each of the modification locations, the inserted instructions having no effect on any output from the execution of the sequence of instructions.

5. The method set forth in claim 4 wherein:

the instructions at the locations specified by the key represent values of digits of the watermark value.

6. The method set forth in claim 1 further comprising the step of:

providing the watermark value to an authenticating entity that authenticates the watermarked code.

7. The method set forth in claim 1 further comprising the step of:

providing the key to the authenticating entity.

8. The method set forth in claim 1 wherein:

the modified sequence of executable instructions is modified such that when the modified sequence of executable instructions is executed, execution state is produced which has a property that depends on the key,

whereby the watermark value is a description of execution state from the modified sequence.

9. The method set forth in claim 8 wherein:

the execution state is a stack depth graph.

10. The method set forth in claim 8 wherein:

the execution state is output from the execution.

11. The method set forth in claim 10 wherein:

the property is an order of elements in the output.

12. The method set forth in claim 10 wherein:

the property is an additional element in the output.

13. The method set forth in claim 10 wherein:

the property is a class of an element in the output.

14. The method set forth in claim 10 wherein:

the property is a constraint that is satisfied by elements of the output.

15. The method set forth in claim 8 further comprising the step of:

providing a description of the produced execution state to an authenticating entity that authenticates the watermarked code.

16. The method set forth in claim 15 further comprising the step of:

providing the key to the authenticating entity.

17. The method set forth in claim 1 further comprising the step of

providing the key to an authenticating entity that authenticates the sequence.

18. A method of authenticating a watermarked sequence of executable instructions, the watermark having been produced by modifying the sequence according to a key such that certain of the instructions in the sequence represent a watermark value,
the method comprising the steps of:

receiving the watermarked sequence or a copy thereof;

using the key to locate the certain instructions in the received sequence and read the watermark value; and

using alteration or absence of the watermark value to determine whether the received sequence is authentic.

19. The method of authenticating set forth in claim 18, the method further comprising the step of:

receiving another watermark value; and

in the step of using alteration or absence of the watermark value to determine whether the received sequence is authentic, the watermark value is compared to the other watermark value.

20. The method of authenticating set forth in claim 19, the method further comprising the step of:

receiving the key.

21. A method of authenticating a sequence of executable instructions that has been watermarked by modifying the sequence according to a key such that when the sequence is executed, first execution state is produced,
the method comprising the steps of:
receiving a description of second execution state; and
if the received description does not describe the first execution state, determining that the sequence of executable instructions whose execution produced the second execution state is not authentic.

22. The method set forth in claim 21 further comprising the step of:
receiving another description of the execution state, the other description describing execution state produced by the execution of the modified sequence; and
in the step of determining, comparing the description and the other description.

23. The method set forth in claim 22 wherein:
the other description is a stack depth graph.

24. The method set forth in claim 21 wherein the execution state is output from the execution, the output having a property which can be determined using the key and
the method further comprises the steps of:
receiving the output from the execution; and
the step of determining includes the steps of
receiving the execution state;
employing the key to determine the property; and
comparing the determined property with the received description.

25. The method set forth in claim 24 wherein:
the determined property is an order of elements in the output.

26. The method set forth in claim 24 wherein:
the determined property is an additional element in the output.

27. The method set forth in claim 24 wherein:

the determined property is a class of an element in the output.

28. The method set forth in claim 24 wherein:

the determined property is a constraint that is satisfied by elements of the output.